Centro Universitário de Excelência
Sistemas de Informação

Separação e estruturação do Sistema via console em Klotin

Autores: João Vitor da Conceição de Almeida
Deivid Souza dos Santos Oliveira Flávio
André Almeida Gomes Neto

Feira de Santana, 2025

Agenda

O objetivo dessa apresentação é apresentar o resultado da refatoração e estruturação do sistema.

1.Contextualização:
Refatoração do código e estruturação do Sistema e implementação de melhorias para manter a legibilidade

2. Descrição da Atividade:
Detalhando a tarefa principal: Estruturação e refatoração de um Sistema de Pedidos em Kotlin via console

3. Estrutura do Sistema Desenvolvido:
Apresentação do menu principal, itens, pedidos e fluxo de funcionamento

Contextualização

- Processo de **revisar e reorganizar o código** sem alterar sua funcionalidade.
- Torna o código **mais limpo, legível e fácil de manter**.
- Reduz complexidade e elimina redundâncias.
- Melhora a clareza e a colaboração entre desenvolvedores.
- Prepara o sistema para **crescimento e novas funcionalidades**.
- Aumenta a consistência e reduz riscos de erros.

Descrição da Atividade

- Melhorar **organização, clareza e escalabilidade** do código.
- Arquitetura **procedural e funcional**, sem orientação a objetos.
- **Separação de responsabilidades**
- **Estruturação dos dados** com `data class`
- **Consistência** com `enum class`
- Código dividido em dois arquivos:
- **Model.kt** → lógica de negócio, `data class`, funções e enum de status.
- **Main.kt** → interação com o usuário, menus e execução do programa.

Estrutura do Sistema

Dividido em dois arquivos.

Model.kt: Nucleo da apresentação

```kotlin
enum class OrderStatus {
    ACCEPTED,
    IN_PROGRESS,
    DONE,
    WAITING_DELIVERY,
    OUT_FOR_DELIVERY,
    DELIVERED
}

data class Item(
    val code: Int,
    val name: String,
    val description: String,
    val price: Double,
    val amount: Int
)

data class Order(
    val code: Int,
    var status: OrderStatus,
    val total: Double,
    val items: MutableList<Item>,
    val hasDiscount: Boolean
)
```

Estrutura do Sistema

Status do pedido:

enum class OrderStatus traz clareza, segurança e padronização.
Ele transforma o código em algo autoexplicativo e menos propenso a erros, o que é essencial numa fase de refatoração.

```
enum class OrderStatus {  10 Usages
    ACCEPTED,   1 Usage
    IN_PROGRESS,
    DONE,
    WAITING_DELIVERY,
    OUT_FOR_DELIVERY,
    DELIVERED
}
```

Separação e estruturação do Sistema via console em Klotin

Estrutura do Sistema

Gerenciamento de itens:

Utiliza Data class para itens do cardápio com funções de nível global para operações como cadastro e atualização. O estado é mantido em uma lista.

```
data class Item(
    val code: Int,
    val name: String,
    val description: String,
    val price: Double,
    val amount: Int
)
```

Estrutura do Sistema

Criação e Gestão de pedidos

```
data class Order(
    val code: Int,
    var status: OrderStatus,
    val total: Double,
    val items: MutableList<Item>,
    val hasDiscount: Boolean
)
```

Pedidos também são modelados com data class, com função de encapsula os dados essenciais de um pedido de forma concisa

Separação e estruturação do Sistema via console em Klotin

Estrutura do Sistema

Separação e estruturação do Sistema via console em Klotin

Estrutura do Sistema

Separação e estruturação do Sistema via console em Klotin

Funcionalidades

- **Funções auxiliares**:
- Para uma experiência de usuário robusta e sem erros, implementamos funções auxiliares de leitura.

readPrice() e readAmount()	Garantem que valores numéricos (preços e quantidades) sejam válidos e maiores que zero ou não-negativos, respectivamente.
readOption() e readExistingCode()	Asseguram que a opção escolhida pelo usuário esteja dentro de um intervalo válido e que o código digitado exista na lista de itens ou pedidos.
readYesNo()	Normaliza a entrada para perguntas "sim/não", aceitando 'Y' ou 'N' e convertendo para booleano.

Considerações finais

- O sucesso do modelo está na manutenção e aplicação de em um estilo procedural e funcional.
- O Principal objetivo foi melhorar a clareza, modularidade e manutenção da lógica de negócio do Sistema Lu-Delivery
- Implementações em Kotlin puro, sem bibliotecas externas
- Base para construção de sistemas de recomendação de alta performance

Resultados

- Todos os Testes passaram

Separação e estruturação do Sistema via console em Klotin